A DIALOGUE GAME FOR DIALECTICAL INTERACTION WITH EXPERT SYSTEMS

T.J.M. Bench-Capon, P.E.S. Dunne and P.H. Leng

Department of Computer Science,
University of Liverpool,
Liverpool,
England

e-mail: ped@uk.ac.liv.csc
Tel: (051)-794-3677; Fax: (051)-794-3715

ABSTRACT

In a variety of applications the most appropriate model of interaction with a knowledge based system is the dialectical discussion, in which a proposition is proposed and argued for. In the course of such a dialogue the tenability of the proposition will be established and the arguments in favour of it adduced and refined. To enable such an interaction it is necessary to formalise the principles underlying such dialogues in a way from which rules to govern system behaviour can be derived. In this paper we argue that the analysis of arguments given by Stephen Toulmin represents a good starting point, and propose a dialogue game based on this schema, adapted to the needs of knowledge based systems. We then sketch an implementation of this game which would provide an interface to a knowledge based system in which queries, their answers and explanations were interleaved in a manner akin to an interpersonal discussion. Such an interface could be used both to interrogate the knowledge base and, if desired, to extend and modify the rules of the knowledge base.

Keywords: Expert Systems; Dialogue; Explanation; Knowledge Acquisition; Dialogue Games

Area: Tools, Techniques and Methods

1. Introduction

When expert systems were first introduced they aspired to very high levels of performance. It was suggested that they could encapsulate all expert knowledge pertaining to a domain and so, when presented with the facts of a particular case, would be able to apply this general expertise and so give advice on that case. Thus the role of the expert system was to store general expertise and the role of the user was to supply specific information to which that expertise would be applied. The result was the consultative model of expert systems, exemplified by MYCIN [3], which was entirely appropriate to such a conception of the use of the system.

As work on expert systems developed, however, two things became apparent. First that the level of expertise required by the expert system to fulfill this role was difficult to attain, and second that decisions in many of the areas to which expert systems were felt to be applicable, such as medicine and law, were not suitable for machine decision. In such areas it is socially important that there is a human being who is ultimately responsible for the decision. Expert systems in such areas thus came to be seen not as oracles which would pronounce their decisions, but as decision support tools to alert the users to some obscure pieces of knowledge of which they might be unaware, and to derive the consequences of the expertise encoded in the knowledge base. This required that the users of the system be themselves competent in the domain, often more competent than the system in some parts of it. Unfortunately this shift, although laudable, and essential for the practical introduction of expert systems into these fields, was not accompanied by a corresponding shift in the mode of interaction. Thus even support systems remained consultative in style, and so the interaction was inappropriate to their role.

For the interaction with an expert support system should not be like the deferential approach to a person whose expertise is regarded as greater than that of the user, but rather like a discussion with a colleague - perhaps a junior colleague - who may provide insight and criticism, but who will in any case provide a way of marshaling and talking through one's ideas. This mismatch between the desired interaction and the interaction offered by a consultative expert system has led to dissatisfaction with expert systems in general, and suggestions for other methods of utilising the kind of knowledge resource represented by an expert system, such as exploration through hypertext. We believe, however, that these problems do not invalidate expert systems: the knowledge base remains a useful knowledge resource and the ability to execute and apply this knowledge has great potential for supporting the user. We have previously argued in [1] that what is required is that a better mode of interaction be provided, so that a session with a supportive expert system will resemble a dialogue of colleagues rather than the questioning of an unsatisfactory expert.

Such an interaction will not be provided simply by restating the output from the expert system in natural language. What is required is a deeper understanding of what is involved in a discussion. This in turn will require the ability to know what is required to present a case for a point of view, to absorb information so presented and to critique cases so presented. Our starting point is that such discussions are highly conventional in nature and so can be seen as being governed by rules. Providing an expert system with these dialogue rules giving information as to the components of such a discussion, how these components are related, and how to re-organise the knowledge and deductions of the expert system in these terms will enable the expert system to engage in this sort of interaction.

Our initial starting point in the search for such rules was the dialogue games of logicians such as MacKenzie [4], discussed in [1]. Our conclusion arising from preliminary work with Mackenzie's game, however, is that the rather formalised dialectical framework it provides is unsatisfactory for the purposes we require. In this paper, therefore, we describe a new dialogue game which uses as its basis an analysis of the structure of arguments derived from the work of Stephen Toulmin [5], which we believe has the richness necessary for the conduct of dialectical interaction with expert systems.

2. Previous Work

MacKenzie's Dialogue Game DC [4] may be characterised as a framework of rules for the conduct of an argument, the purpose of which is to arrive at a conclusion with which both participants agree.

The rules of DC are,in particular, designed to ensure that participants accept the logical consequences of statements which they accept to be true, and, importantly, to avoid circular arguments with the attendant fallacy of 'begging the question'.

Two aspects of MacKenzie's game attracted our interest in considering ways of interacting with a KBS. The first was that DC defines a dialogue between equal participants: either may contribute information which may or may not be accepted by the other. It appeared to us that this kind of dialogue offered a basis for a more intelligent model of human-computer interaction than that assumed by a one-sided consultative system. The other interesting property of DC is that it is a non-cumulative system: that is, in the course of an argument, either participant may withdraw commitments previously made to the truth of statements. The implication of this is that a participant may, as part of an argument, advance a proposition the truth of which is in doubt. On being confronted with an inconsistency arising from the commitment to this proposition, he or she may withdraw the commitment and either accept the conclusion or continue the argument in different ways. Again, it was our view that this property provides a more accurate and useful representation of the way in which real-life dialogues proceed between participants who may each contribute to the discussion a mixture of hard facts and more tentative suppositions which may, nevertheless, be helpful in developing the argument.

In [1] we described a Dialogue Abstract Machine (DAM) the purpose of which was to mediate in a dialogue conducted according to the rules of MacKenzie's game, and we have subsequently carried out a preliminary implementation of this [6]. Unsurprisingly, this exercise has exposed some of the limitations of MacKenzie's game for the purpose we have in mind. The rules of the game encourage the development of linear chains of commitment and implication, in which each link has equal status. As a strictly logical demonstration of the truth of a conclusion this is adequate; in informal, "real-life" debate, however, the assertions made in support of a conclusion do not necessarily all have equal weights or identical roles, and this strictly linear argument structure is unhelpful in identifying those premises which are of genuine significance in the development of the argument. Furthermore, in any but a trivial argument, this linearity rapidly leads to very clumsy and complex verbalisations in the dialogue; especially, demands of the form "Resolve whether "*A*" and "*B*" and "*A* and *B* implies *C*" implies "*C*" is true" (of which this is in fact a very simple example).

MacKenzie's game, as is only to be expected given that his motive for producing the game was to explore fallacies in logical argument, sees discussion as the production of a formal proof, and the moves of the game are the moves that are made by formal logicians producing such a proof. We have argued elsewhere [2], that one reason why the explanations of expert systems are so unsatisfactory is that they present a proof when what is required is an argument. An argument differs from a proof in that it uses differences amongst the premises in terms of their importance and their role in establishing the point under discussion to organise and present the material in a way which the interlocutor will find persuasive and convincing. Similarly we have come to believe that any dialogue game suitable as a basis for a dialectical interaction with an expert system will be founded on the notion of an argument rather than a logical proof.

These reasons have led us to investigate the analysis of arguments proposed by Toulmin [5]. The argument schema that he developed was used as a basis for the explanations described in [2], and is here adapted to the requirements of a dialogue game for human-KBS interaction. The next section gives a brief introduction to Toulmin's analysis of arguments.

3. Toulmin's Analysis of Arguments

Toulmin's starting point was that arguments are not best illuminated by being cast into the mould of premises and conclusions. Instead he suggested a division into a more varied set of components that would highlight the role of the various kinds of assertions made in the course of an argument. The structure suggested by Toulmin may be represented diagrammatically, as in Figure 1.

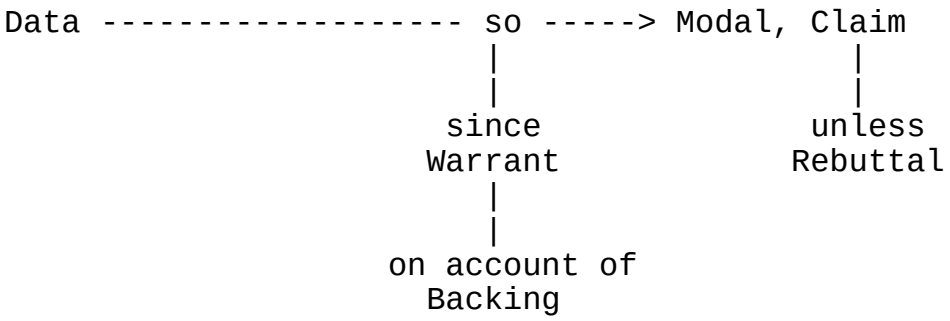

Figure 1

The arguments are decomposed into *claims* whose truth we seek to establish by the argument, *data* that we appeal to as the grounds for the claim, and *warrants* which provide the rules of inference that connect the data and the claim. Warrants can, in Toulmin's scheme, bestow varying degrees of support for the claim, and these degrees of support are indicated by the use of a modal *qualifier,* such as "necessarily" or "possibly". There may also be exceptional conditions which prevent the claim from being established: these are indicated by the *rebuttal* which contains circumstances acknowledged as requiring the authority of the warrant to be put aside. Warrants also require some justification: this is the role of the *backing.*

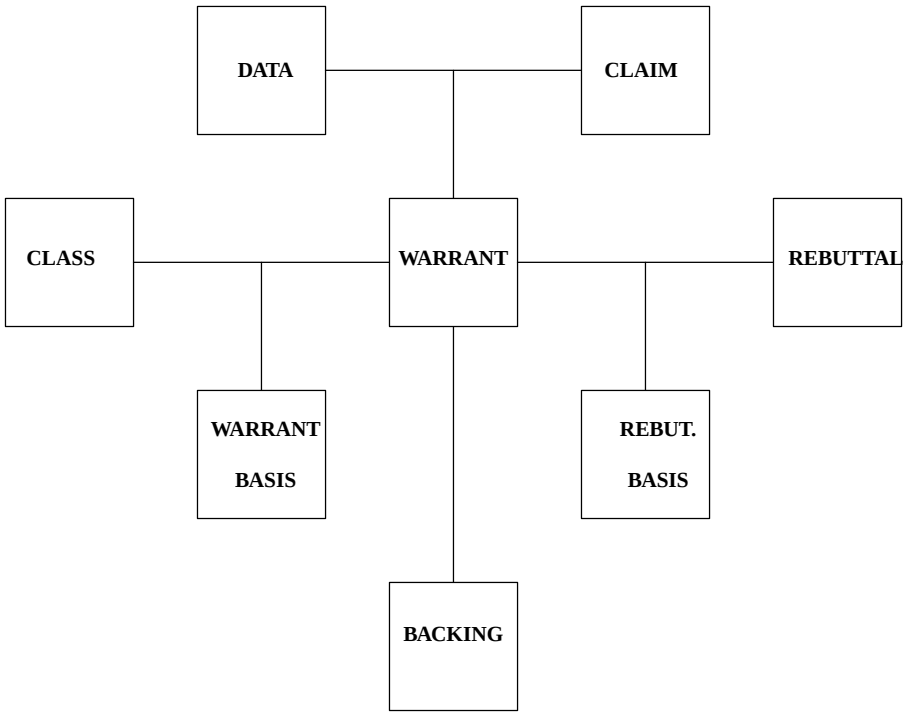

Figure 2

In applying this schema to logic programs in [2] we dropped the modal qualifier, and extended the schema by the inclusion of a further component which was designed to determine the applicability of the warrant, which we termed the *class* as it indicated the class of things to which the warrant applied. The grounds for this inference from class to applicability of warrants were provided by a further component called the *basis.* For our present purposes we have extended the schema further in order to provide a basis for the rebuttal also. This means that the rebuttal had to be located as preventing the application of the warrant, rather than an immediate refutation of the claim. We provide for the possibility of a chain of arguments by permitting components of the schema to be the claims

of previous arguments. The modified schema thus appears as in Figure 2.

As an example of the kind of dialogue which we can analyse in this way, consider the following argument concerning the sterilisation of surgical implements. We will suppose that a particular kind of germ has been identified, and the question is posed:

Questioner: Will boiling water kill these germs?

Expert: No. *This becomes the claim, made because the expert knows that these germs survive temperatures of up to 103°.*.

Q: Why not? *The answer was not what was expected, so explanation is sought .*

E: The germs die at 103^{O}C. *The reply given is the most important reason for the claim, i.e the data.*

Q: And? *The questioner is still not convinced.*

E: Water boils at 100°. *The other premise - not given first because Q was expected to know this.*

Q: But the water was boiled in Portsmouth. *The questioner detects the reasoning flaw and sets up a rebuttal.*

E: So? *The expert knows nothing to make this new fact relevant .*

Q: Portsmouth water boils at 104°. *This is the basis for the rebuttal.*

E: Why? *This contradicts what the expert believes, so justification is required.*

Q: It contains a lot of chalk *This is the other item of data required to establish the rebuttal.*

E: So? *The expert now seeks the rule which led to the inference.*

Q: Chalky water boils at 104°.

Figure 3 illustrates how this dialogue may be represented using our modified form of the Toulmin schema. Notice in particular how the basis for the rebuttal of the original claim has, when challenged by E, itself become the claim of a subsidiary argument. The example also illustrates how a non-expert may sometimes be in a position to contribute significant specialised information in the course of a dialogue; Q is not an expert on the survival of bacteria, but does happen to know a curious "fact" about water in Portsmouth.

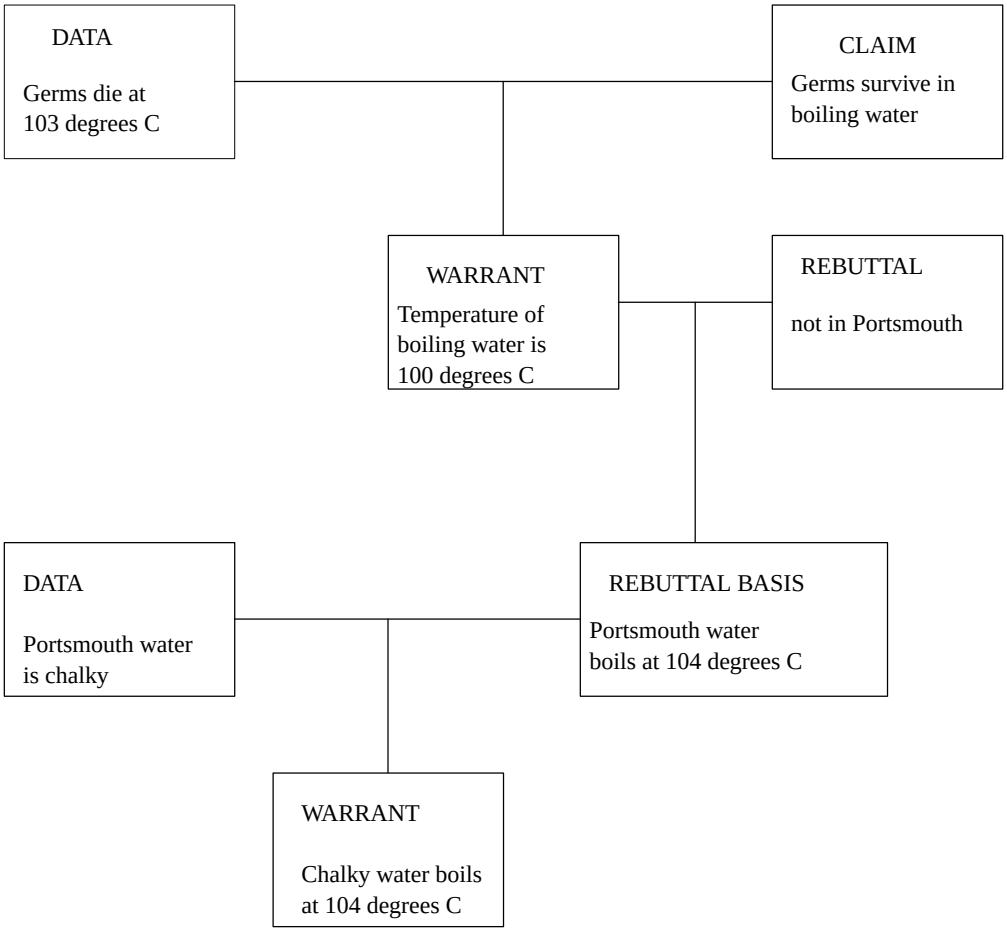

Figure 3

4. The Toulmin Dialogue Game (TDC)

In this section we outline the elements of a formal dialogue game, TDC, based on our extension of Toulmin's argument schema. While the game embodies many of the elements of MacKenzie's game DC it is more concerned with providing a basis for developing arguments and explanations in dialogues than with the avoidance of logical fallacies such as question-begging. As such we do not give a full formal breakdown of the dialogue rules in TDC *a la* MacKenzie but concentrate instead on how the components of TDC are regulated in a dialogue.

TDC utilises six locution types: *assertions* and *rules* which correspond to the type *statement* in DC; *questions* ('*is it the case that s*'); *challenges* ('*Why s?*'); *resolution demands* ('*Resolve whether s*'); and *withdrawals* ('*No commitment s*').

The locution type *rule* falls into two subtypes.

i) *warrants* and *bases*: These are rules of the form '*c if k*' where *c* is an assertion locution and *k* a conjunction of *annotated* assertions.

ii) *rebuttal bases*: These are rules of the form '$\neg$ *c if k*' where *c* and *k* are as above.

The annotation associated with a rule is to correspond with the labelling used in Toulmin's schema.

One important element of DC is the use of a *commitment store* by each protagonist to record which statements are accepted and under challenge. As we observed in [1], this is basically a simple linear structure. In TDC a more complex mechanism is used in order to facilitate the processes of argument and explanation. The basic element of this structure is called a *Toulmin graph*.

Definition 1: A *skeletal Toulmin graph* is a labelled directed graph, $T(V, E)$. Each node of the graph is labelled with (the name of) an argument and each edge is labelled to show which component of the destination argument is supported by the source argument, e.g. if an edge from argument *P* to

argument Q is labelled '*data-for*' then this indicates that P provides the data node of the schema for the argument Q. An *expanded* Toulmin graph is a skeletal Toulmin graph in which some nodes have been replaced by a subgraph of the structure of Figure 2. A *fully expanded* Toulmin graph is one in which all argument nodes have been so replaced. It should be noted that there is no requirement for a Toulmin graph to be topologically connected, i.e. the graph may contain several non-interacting argument networks. •

Definition 2: Let B and W denote a pair of protagonists; S denote a set of all statements (where a statement is a rule or a conjunction of assertions). L the set of all locutions is

$$L = \{ s, Q's, R's, W's, Y's : s \in S \}$$

i.e. the set of all statements, questions, resolution demands, withdrawals and challenges. A *locution act*, l, is any element of $\{B, W\} \times L$. A *dialogue* (of *length* n) is a sequence of n pairs $< i, l >$ where $1 \leq i \leq n$ and l is a locution act. Each protagonist, X, maintains a *commitment graph*, $CG(X)$, which is an expanded Toulmin graph. A *dialogue game* over L is defined by a *legal move predicate*, Π; and a *commitment function* C. The legal move predicate defines which locution acts may extend a dialogue of length n to a dialogue of length $n+1$, taking into account the dialogue so far and the current commitment graph of each protagonist. The commitment function states how $CG(X)$ is to be modified after the nth locution act where $n \geq 1$. •

Before examining the legal move and commitment function we note that there will be one important difference between TDC and MacKenzie's game: each locution act may effect a change in the commitment graph of each protagonist; we thus equate different types of locution with corresponding Toulmin graph *operations*. In this way, since at any stage the currently applicable operations depend on the current commitment graphs, it is sufficient to define *only* the commitment function.

The operations that can be performed on a Toulmin graph (and their corresponding locution classes) are as follows:

Create: Create a new argument node. In effect this corresponds to the locution type 'statement'.

Expand: Replace an argument node by its expanded form. This is the response required to a challenge locution.

Grow: Add a missing node (possibly unfilled) to an expanded argument schema.

Fill: Place a statement within a node of an expanded schema

The *grow* and *fill* operations constitute changes arising from statements following challenges. Thus where the expanded form of an argument is a single claim node, the challenger may, for example, request that 'data' or a 'warrant' be supplied for the claim. The *fill* operation is used to instantiate (or modify) a schema node. Note that the *grow* operation must leave a legal Toulmin schema and so an attempt to grow a rebuttal for a claim with no warrant would not be permitted.

Collapse: This replaces an expanded schema by a single argument node in the Toulmin graph. An unexpanded argument node is taken as representing a statement which is committed to. Thus collapsing a schema is the effect of a positive response to a resolution demand.

Delete: Either an argument node or a schema node may be deleted. This operation is equivalent to withdrawing any commitment to a statement.

With this approach, TDC is formally viewed as a game played on particular types of directed graph, with the moves (graph operations) mirroring specific dialogue actions. As a more detailed example, consider a challenge such as '*Why c?*' and its corresponding treatment as a series of graph operations. Combining Toulmin's argument formalism with DC we can summarise the basic responses as:

- <c> since <Data>
- <c> since <Warrant {and backing}> and ¬ <rebuttal>
- <c> since <Warrant {and backing}>

- ¬ <c> since (<Rebuttal>)
- No commitment <c>

where the phrase '{and backing}' is optional in the locution. If present it provides an explanation of the warrant using appropriate textual evidence. Each of the components — Data, Warrant, Rebuttal — can themselves be subsequently challenged. In terms of operations on a Toulmin graph these responses can be treated in a much richer fashion as follows.

The challenger of 'c' must have the argument node for 'c' in its expanded form. Since 'c' *itself* is under challenge there can be no substantiating data or warrant node in this schema (otherwise grow and fill requests could be issued to obtain these rather than the challenge on 'c'). The hearer of the challenge examines his commitment graph and if there is an argument node corresponding to 'c' expands this. It should be noted that since 'c' could not be currently under challenge, if present the argument node must be in its collapsed form. The structure of the schema can be used to determine the hearer's response. Thus data, warrant or rebuttal concerning 'c' can be supplied by the corresponding *create* operation. The challenger, if not satisfied with this, may issue further challenges to these or may request data for a warrant (or vice-versa). The chain of challenges may end with a resolution demand that has the effect of collapsing expanded schemata in the argument process.

5. Implementation

The storage and execution model assumed by a computer system for mediating in a rule-based dialogue may be described conveniently as an abstract machine, the execution semantics of which enforce the rules of the game, and the state of which at any moment defines the current state of the dialogue in progress. In the case of the Dialogue Abstract Machine described earlier in [1], for MacKenzie's game, the state components included both a commitment store and a dialogue graph, the latter to retain a history of the steps in the dialogue. For TDC, however, it is sufficient to record the state in the form of the commitment graph introduced in the previous section; for convenience, commitments of both participants can be recorded within a single graph structure. The representation of this graph forms the basic memory structure of our abstract machine, which we will call TDAM.

Execution steps in the progress of a dialogue involve, in general, traversal of and modifications and extensions to this graph structure. Although TDC is essentially a dialogue between equal participants, we expect that in practice one of these (*P*) will be a human participant who will be the initiator of the dialogue, while the other (*K*) will be a Knowledge Based System whose role will be largely responsive. The usual starting point in a dialogue will be a claim by *P*, representing an assertion the truth of which is to be examined. The response in TDAM to this will be to create in its memory an expanded Toulmin node structure, of the form illustrated in Figure 2, to represent the assertions and rules which may be produced to verify (or disprove) this claim. This is followed by an invitation to *K* to furnish this structure with information pertaining to the claim which it may (or may not) be able to produce from its knowledge base.

The subsequent progress of the dialogue will depend on the extent of the information which *K* can bring to bear on the problem; the willingness (or otherwise) of *P* to accept this information without further argument; and the ability of *P* to contribute additional information. For example, *P* may doubt the data produced by *K* in support of the claim, and may express this doubt by issuing a challenge to the data component in the current argument schema. The effect of this will be to treat this data as a claim by *K*, and to extend the commitment graph by adding a structure representing the antecedents of this claim, for which *K* will be invited to provide further information.

Conversely, *K* may be unable to verify a claim made by *P*, in which case *P* will be called upon to provide information, in the form of assertions and/or rules, which will be stored in the argument structure created for the claim in the commitment graph. *K* may in turn respond by challenging assertions made by *P*, or (if it is able to do so) providing further information to support or refute these assertions.

In order to sustain dialogues of this form, TDAM requires, in addition to the commitment-graph memory and the context-sensitive execution semantics which implement the rules of TDC, means of interacting with both the human and KBS participants. Interaction with the KBS takes the form of normal enquiries, the responses to which are translated into the required Toulmin schema for incorporation in the commitment graph. Alternative modes of interaction may be permitted to produce either conservative responses (in which each predicate must be requested separately) or full responses, in which the KBS will respond to a challenge by attempting to furnish information for the entire Toulmin schema.

The merits we see in conducting a dialogue using this framework emerge most clearly in the human-computer interaction. Use of the schema defined in Figure 2 as a basis for representing each stage in the dialogue lends itself naturally to an implementation which makes use of windows and menu selection rather than textual constructions. We envisage a dialogue step, on the part of the human participant, as taking the form of a selection of a node in a graphical representation of the argument schema, followed by selection of a dialogue option chosen from a (context-dependent) menu. The response of TDAM may, in appropriate cases, involve the opening of a window (representing an expansion of a node) or the return to a previous structure following the satisfactory resolution of a subsidiary dialogue. The advantages we see in this kind of interaction are that it allows the user to choose to explore, by navigation through the expanded Toulmin graph, those parts of an argument which are of particular interest, ignoring others whose truth may not be in dispute, and that the graphical format avoids the textual complexity which tends to emerge in other forms of dialogue.

A further possibility is opened up when we consider the state of the dialogue at the end of the session. At that point a large structure will exist representing knowledge supplied both by the system and by the user. If we wish we may use the knowledge represented in this form to extend or modify the system's knowledge base. To do this would mean forcing the system to accept what was supplied by the user: given a sufficiently authoritative user, however, this seems not unreasonable. Now we can reverse the process by which the system produced the argument components and map the structure back into system rules. Thus, for example, unsupported data supplied by the user will become facts, warrants supplied by the user will become rules, annotated according to the surrounding structure, and rebuttals given by the user will modify existing rules in the knowledge base by adding in the appropriate extra conditions. This facility allows the system to "learn" from the dialogues, and suggests a novel means of knowledge acquisition that should appeal to the domain expert for whom such explanatory discussion will be a natural mode of discourse.

6. Conclusions

Dialogue games appear to offer a useful model for human interaction with a knowledge-based system. The attraction of a game such as MacKenzie's is that it provides a precise and formal set of rules for the conduct of a dialogue, while allowing within this framework freedom for the human participant to question the KBS, to disagree with its assertions, and to contribute additional information to the discussion. The weakness of MacKenzie's game in particular, however, is that it is based too strongly on the concept of formal proof, leading to dialogues which are both excessively pedantic and linguistically clumsy and over-complex. This has led us to consider the argument schema described by Toulmin as an alternative basis for the conduct of a rule-based dialogue. We have used a modified version of the Toulmin schema as a structural framework on which to develop a dialogue game, based loosely on the concepts of MacKenzie's game, and to describe an implementation for use in human-computer interaction.

The principal advantage of using this approach is that it provides a means of defining the context within which an assertion appears in the course of a dialogue, and of representing this information graphically, avoiding the need for cumbersome verbalisations. The structural information embodied in the argument schemata augments the textual detail incorporated in the nodes of the structure to provide a clearer form of explanation of the conclusions reached. The dialogue-game concept of 'commitment', and the non-cumulative nature of the game, also allows for the expression of uncertainty in the argument: a conclusion reached following the resolution of an argument need

not be absolute, but may be subject to the provisional acceptance of assertions to which either or both participants remain uncommitted, or which may subsequently be withdrawn. The model of a dialogue game, and the structure provided by the argument schema, offers a suitable framework not only for the contribution of user-supplied information to the dialogue in this way, but also for this to be incorporated into the knowledge base if it is subsequently wished that this be done.

7. References

[1] Bench-Capon, T.J.M., Dunne, P.E.S., and Leng, P.H., *Interacting With Knowledge Based Systems Through Dialogue Games*, 11th Annual Conference on Expert Systems and Their Applications, Avignon, 1991, pp123-130.

[2] Bench-Capon, T.J.M., Lowes, D, and McEnery, A.M., *Using Toulmin's Argument Schema to Explain Logic Programs.* Knowledge Based Systems, Vol 4, No 3, pp 177-183, 1991.

[3] Buchanan, B.G. and Shortliffe E.H., *Rule-Based Expert Systems*, Addison Wesley, 1984.

[4] MacKenzie, J.D., *Question-Begging in Non-Cumulative Systems*, Journal of Philosophical Logic, vol 8, 1979, pp 159-177.

[5] Toulmin, S., *The Uses of Argument*, Cambridge University Press, 1958.

[6] Leung, G., *Implementation of a Dialogue Abstract Machine*, MSc Dissertation, University of Liverpool, 1991.